

Burn

Generated by Doxygen 1.8.11

Contents

1	burn	1
2	Namespace Index	3
2.1	Packages	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	burn Namespace Reference	11
6.2	burn.burn Namespace Reference	11
6.2.1	Variable Documentation	11
6.2.1.1	filename	11
6.2.1.2	level	11
6.3	burn.helpers Namespace Reference	12
6.3.1	Function Documentation	12
6.3.1.1	setblocking(fd, state)	12
6.4	burn.net_proc Namespace Reference	12
6.4.1	Variable Documentation	12
6.4.1.1	HOST	12
6.4.1.2	PORT	12
6.5	burn.proto Namespace Reference	13
6.6	burn.spec_proc Namespace Reference	13

7	Class Documentation	15
7.1	burn.burn.Burn Class Reference	15
7.1.1	Detailed Description	15
7.1.2	Constructor & Destructor Documentation	15
7.1.2.1	__init__(self)	15
7.1.3	Member Function Documentation	16
7.1.3.1	__enter__(self)	16
7.1.3.2	__exit__(self, exc_type, exc_value, traceback)	16
7.1.3.3	dispatch_net_msg(self, msg)	16
7.1.3.4	dispatch_spec_msg(self, msg)	17
7.1.3.5	run(self)	17
7.1.4	Member Data Documentation	17
7.1.4.1	fdn	17
7.1.4.2	fds	18
7.1.4.3	n	18
7.1.4.4	running	18
7.1.4.5	s	18
7.2	burn.spec_proc.GpsThread Class Reference	18
7.2.1	Detailed Description	19
7.2.2	Constructor & Destructor Documentation	19
7.2.2.1	__init__(self, event)	19
7.2.3	Member Function Documentation	19
7.2.3.1	altitude(self)	19
7.2.3.2	eps(self)	20
7.2.3.3	epv(self)	20
7.2.3.4	epx(self)	20
7.2.3.5	epy(self)	20
7.2.3.6	latitude(self)	20
7.2.3.7	longitude(self)	20
7.2.3.8	run(self)	21

7.2.3.9	speed(self)	21
7.2.3.10	time(self)	21
7.2.4	Member Data Documentation	21
7.2.4.1	gpsd	21
7.2.4.2	last_alt	22
7.2.4.3	last_eps	22
7.2.4.4	last_epv	22
7.2.4.5	last_epx	22
7.2.4.6	last_epy	22
7.2.4.7	last_lat	22
7.2.4.8	last_lon	22
7.2.4.9	last_speed	22
7.2.4.10	last_time	22
7.3	burn.proto.Message Class Reference	23
7.3.1	Detailed Description	23
7.3.2	Constructor & Destructor Documentation	23
7.3.2.1	__init__(self, command="", arguments={})	23
7.3.3	Member Data Documentation	23
7.3.3.1	arguments	23
7.3.3.2	command	24
7.4	burn.net_proc.NetProc Class Reference	24
7.4.1	Detailed Description	24
7.4.2	Constructor & Destructor Documentation	25
7.4.2.1	__init__(self, fd)	25
7.4.3	Member Function Documentation	25
7.4.3.1	dispatch_ctrl_msg(self, msg)	25
7.4.3.2	dispatch_net_msg(self)	26
7.4.3.3	is_running(self)	26
7.4.3.4	run(self)	27
7.4.4	Member Data Documentation	27

7.4.4.1	addr	27
7.4.4.2	buffer	27
7.4.4.3	fd	28
7.4.4.4	sock	28
7.5	object Class Reference	28
7.6	burn.spec_proc.SessionThread Class Reference	28
7.6.1	Detailed Description	28
7.6.2	Constructor & Destructor Documentation	29
7.6.2.1	__init__(self, event, target, msg)	29
7.6.3	Member Function Documentation	29
7.6.3.1	run(self)	29
7.7	burn.spec_proc.SpecProc Class Reference	30
7.7.1	Detailed Description	30
7.7.2	Constructor & Destructor Documentation	31
7.7.2.1	__init__(self, fd)	31
7.7.3	Member Function Documentation	31
7.7.3.1	dispatch(self, msg)	31
7.7.3.2	reset_acquisition(self)	32
7.7.3.3	run(self)	32
7.7.3.4	run_acquisition(self, msg, session_index)	33
7.7.3.5	run_acquisition_once(self, req_msg, session_index)	33
7.7.3.6	save_acquisition(self, msg, session_index)	34
7.7.3.7	save_acquisition_as_chn(self, sd, msg, session_index)	35
7.7.3.8	send_msg(self, msg)	35
7.7.3.9	stabilize_probe(self, voltage, coarse_gain, fine_gain)	36
7.7.4	Member Data Documentation	36
7.7.4.1	dtb	36
7.7.4.2	fd	36
7.7.4.3	gps_client	36
7.7.4.4	gps_stop	36
7.7.4.5	group	37
7.7.4.6	input	37
7.7.4.7	running	37
7.7.4.8	send_lock	37
7.7.4.9	session	37
7.7.4.10	session_stop	37

8 File Documentation	39
8.1 /home/drb/dev/py/burn/__init__.py File Reference	39
8.2 /home/drb/dev/py/burn/burn.py File Reference	39
8.3 /home/drb/dev/py/burn/helpers.py File Reference	39
8.4 /home/drb/dev/py/burn/net_proc.py File Reference	40
8.5 /home/drb/dev/py/burn/proto.py File Reference	40
8.6 /home/drb/dev/py/burn/README.md File Reference	40
8.7 /home/drb/dev/py/burn/spec_proc.py File Reference	40
Index	41

Chapter 1

burn

Daemon running on Raspberry Pi (archlinuxarm), controlling a Canberra Osprey gamma detector and a G-Star IV GPS device.

Acquisitions and measurements are merged, saved to disk and optionally transferred over a TCP connection to the controlling application, crash.

Dependencies:

- Canberra Osprey SDK V1.0.1

This software is part of a drone project at Norwegian Radiation Protection Authority (NRPA)

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

burn	11
burn.burn	11
burn.helpers	12
burn.net_proc	12
burn.proto	13
burn.spec_proc	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

burn.burn.Burn	15
object	28
burn.proto.Message	23
Thread	
burn.spec_proc.GpsThread	18
burn.spec_proc.SessionThread	28
Process	
burn.net_proc.NetProc	24
burn.spec_proc.SpecProc	30

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

burn.burn.Burn	15
burn.spec_proc.GpsThread	18
burn.proto.Message	23
burn.net_proc.NetProc	24
object	28
burn.spec_proc.SessionThread	28
burn.spec_proc.SpecProc	30

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/home/drb/dev/py/burn/ __init__.py	39
/home/drb/dev/py/burn/ burn.py	39
/home/drb/dev/py/burn/ helpers.py	39
/home/drb/dev/py/burn/ net_proc.py	40
/home/drb/dev/py/burn/ proto.py	40
/home/drb/dev/py/burn/ spec_proc.py	40

Chapter 6

Namespace Documentation

6.1 burn Namespace Reference

Namespaces

- [burn](#)
- [helpers](#)
- [net_proc](#)
- [proto](#)
- [spec_proc](#)

6.2 burn.burn Namespace Reference

Classes

- class [Burn](#)

Variables

- [filename](#)
- [level](#)

6.2.1 Variable Documentation

6.2.1.1 burn.burn.filename

Definition at line 125 of file burn.py.

6.2.1.2 burn.burn.level

Definition at line 125 of file burn.py.

6.3 burn.helpers Namespace Reference

Functions

- def [setblocking](#) (fd, state)

6.3.1 Function Documentation

6.3.1.1 def burn.helpers.setblocking (fd, state)

Set the blocking state of a file descriptor

Definition at line 22 of file helpers.py.

```
22 def setblocking(fd, state):
23     """
24     Set the blocking state of a file descriptor
25     """
26     flags = fcntl.fcntl(fd, fcntl.F_GETFL)
27     if state:
28         fcntl.fcntl(fd, fcntl.F_SETFL, flags & ~os.O_NONBLOCK)
29     else:
30         fcntl.fcntl(fd, fcntl.F_SETFL, flags | os.O_NONBLOCK)
31
```

6.4 burn.net_proc Namespace Reference

Classes

- class [NetProc](#)

Variables

- string [HOST](#) = "
- int [PORT](#) = 7000

6.4.1 Variable Documentation

6.4.1.1 string burn.net_proc.HOST = "

Definition at line 25 of file net_proc.py.

6.4.1.2 int burn.net_proc.PORT = 7000

Definition at line 26 of file net_proc.py.

6.5 burn.proto Namespace Reference

Classes

- class [Message](#)

6.6 burn.spec_proc Namespace Reference

Classes

- class [GpsThread](#)
- class [SessionThread](#)
- class [SpecProc](#)

Chapter 7

Class Documentation

7.1 burn.burn.Burn Class Reference

Public Member Functions

- def `__init__` (self)
- def `run` (self)
- def `dispatch_net_msg` (self, msg)
- def `dispatch_spec_msg` (self, msg)
- def `__enter__` (self)
- def `__exit__` (self, exc_type, exc_value, traceback)

Public Attributes

- `running`
- `fds`
- `fdn`
- `s`
- `n`

7.1.1 Detailed Description

Definition at line 30 of file burn.py.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def burn.burn.Burn.__init__ (self)

```
initialize main controller
```

Definition at line 32 of file burn.py.

```

32     def __init__(self):
33         """
34         initialize main controller
35         """
36         self.running = False
37
38         # Create pipes for message passing between gps_proc and spec_proc
39         fds_pass, self.fds = Pipe()
40         fdn_pass, self.fdn = Pipe()
41
42         # Make file descriptors non-blocking
43         setblocking(self.fds, 0)
44         setblocking(self.fdn, 0)
45
46         # Create and start child processes
47         self.s = SpecProc(fds_pass)
48         self.n = NetProc(fdn_pass)
49         self.s.start()
50         self.n.start()
51
52         # Close unused file descriptors
53         # The child processes inherits *all* parent descriptors (FIXME)
54         fds_pass.close()
55         fdn_pass.close()
56

```

7.1.3 Member Function Documentation

7.1.3.1 def burn.burn.Burn.__enter__(self)

Definition at line 106 of file burn.py.

```

106     def __enter__(self):
107         return self
108

```

7.1.3.2 def burn.burn.Burn.__exit__(self, exc_type, exc_value, traceback)

Definition at line 109 of file burn.py.

```

109     def __exit__(self, exc_type, exc_value, traceback):
110         # Main controller destructor, cleaning up
111         self.fds.close()
112         self.fdn.close()
113
114         self.s.join()
115         self.n.join()
116
117         logging.info('ctrl: terminating')
118

```

7.1.3.3 def burn.burn.Burn.dispatch_net_msg(self, msg)

Function to handle messages from the network

Definition at line 81 of file burn.py.


```

81     def dispatch_net_msg(self, msg):
82         """
83         Function to handle messages from the network
84         """
85         if not msg:
86             return
87         if msg.command == 'ping':
88             msg.command = 'ping_ok'
89             self.fdn.send(msg)
90         elif msg.command == 'close':
91             # Ground control has requested a close down
92             self.fds.send(msg) # Notify spectrometer
93             msg.command = 'close_ok' # Convert the message to a response
94             self.fdn.send(msg) # Notify network process and ground control
95             self.running = False
96         elif msg.command == 'new_session' or msg.command == 'stop_session' or msg.command == 'set_gain':
97             self.fds.send(msg) # No housekeeping necessary, pass directly to spectrometer
98         else:
99             # Unknown command received from network
100             logging.warning('ctrl: unknown command: ' + msg.command)
101

```

7.1.3.4 def burn.burn.Burn.dispatch_spec_msg (self, msg)

Definition at line 102 of file burn.py.

```

102     def dispatch_spec_msg(self, msg):
103         # Message received from spectrometer, pass on to network
104         self.fdn.send(msg)
105

```

7.1.3.5 def burn.burn.Burn.run (self)

Entry point for the main controller

Definition at line 57 of file burn.py.

```

57     def run(self):
58         """
59         Entry point for the main controller
60         """
61         self.running = True
62
63         # Wait for gps to start up
64         logging.info('ctrl: warming up services')
65         time.sleep(5)
66
67         # Prepare file descriptors for selection
68         inputs = [self.fdn, self.fds]
69
70         # Main event loop
71         while self.running:
72             readable, _, _ = select.select(inputs, [], [])
73             # Process readable file descriptors
74             for s in readable:
75                 msg = s.recv()
76                 if s is self.fdn: # Message received from network
77                     self.dispatch_net_msg(msg) # Handle network message
78                 elif s is self.fds: # Message received from spectrometer
79                     self.dispatch_spec_msg(msg) # Handle spectrometer message
80

```

7.1.4 Member Data Documentation

7.1.4.1 burn.burn.Burn.fdn

Definition at line 40 of file burn.py.

7.1.4.2 burn.burn.Burn.fds

Definition at line 39 of file burn.py.

7.1.4.3 burn.burn.Burn.n

Definition at line 48 of file burn.py.

7.1.4.4 burn.burn.Burn.running

Definition at line 36 of file burn.py.

7.1.4.5 burn.burn.Burn.s

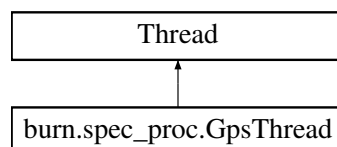
Definition at line 47 of file burn.py.

The documentation for this class was generated from the following file:

- </home/drb/dev/py/burn/burn.py>

7.2 burn.spec_proc.GpsThread Class Reference

Inheritance diagram for burn.spec_proc.GpsThread:



Public Member Functions

- `def __init__ (self, event)`
- `def run (self)`
- `def latitude (self)`
- `def epx (self)`
- `def longitude (self)`
- `def epy (self)`
- `def altitude (self)`
- `def epv (self)`
- `def speed (self)`
- `def eps (self)`
- `def time (self)`

Public Attributes

- [gpsd](#)
- [last_lat](#)
- [last_epx](#)
- [last_lon](#)
- [last_epy](#)
- [last_alt](#)
- [last_epv](#)
- [last_speed](#)
- [last_eps](#)
- [last_time](#)

7.2.1 Detailed Description

Thread class to handle the gps driver

Definition at line 38 of file spec_proc.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 def burn.spec_proc.GpsThread.__init__(self, event)

Description:

Initialize the gps thread

Definition at line 42 of file spec_proc.py.

```

42     def __init__(self, event):
43         """
44         Description:
45             Initialize the gps thread
46         """
47         threading.Thread.__init__(self)
48         self.__stopped = event
49         self.gpsd = gps(mode=WATCH_ENABLE)
50         self.last_lat = 0
51         self.last_epx = 0
52         self.last_lon = 0
53         self.last_epy = 0
54         self.last_alt = 0
55         self.last_epv = 0
56         self.last_speed = 0
57         self.last_eps = 0
58         self.last_time = ''
59

```

7.2.3 Member Function Documentation

7.2.3.1 def burn.spec_proc.GpsThread.altitude(self)

Definition at line 111 of file spec_proc.py.

```

111     def altitude(self):
112         return self.last_alt
113

```

7.2.3.2 def burn.spec_proc.GpsThread.eps (self)

Definition at line 123 of file spec_proc.py.

```
123     def eps(self):
124         return self.last_eps
125
```

7.2.3.3 def burn.spec_proc.GpsThread.epv (self)

Definition at line 115 of file spec_proc.py.

```
115     def epv(self):
116         return self.last_epv
117
```

7.2.3.4 def burn.spec_proc.GpsThread.epx (self)

Definition at line 99 of file spec_proc.py.

```
99     def epx(self):
100         return self.last_epx
101
```

7.2.3.5 def burn.spec_proc.GpsThread.epy (self)

Definition at line 107 of file spec_proc.py.

```
107     def epy(self):
108         return self.last_epy
109
```

7.2.3.6 def burn.spec_proc.GpsThread.latitude (self)

Definition at line 95 of file spec_proc.py.

```
95     def latitude(self):
96         return self.last_lat
97
```

7.2.3.7 def burn.spec_proc.GpsThread.longitude (self)

Definition at line 103 of file spec_proc.py.

```
103     def longitude(self):
104         return self.last_lon
105
```

7.2.3.8 def burn.spec_proc.GpsThread.run (self)

Description:

Entry point for the gps thread

Definition at line 60 of file spec_proc.py.

```

60     def run(self):
61         """
62         Description:
63             Entry point for the gps thread
64         """
65         logging.info('gps: starting service')
66
67         # Process any buffered gps signals every .3 seconds
68         while not self._stopped.wait(0.3):
69
70             # Update our last measurement until buffer is empty
71             while self.gpsd.waiting():
72                 self.gpsd.next()
73                 if not math.isnan(self.gpsd.fix.latitude):
74                     self.last_lat = self.gpsd.fix.latitude
75                 if not math.isnan(self.gpsd.fix.epx):
76                     self.last_epx = self.gpsd.fix.epx
77                 if not math.isnan(self.gpsd.fix.longitude):
78                     self.last_lon = self.gpsd.fix.longitude
79                 if not math.isnan(self.gpsd.fix.epy):
80                     self.last_epy = self.gpsd.fix.epy
81                 if not math.isnan(self.gpsd.fix.altitude):
82                     self.last_alt = self.gpsd.fix.altitude
83                 if not math.isnan(self.gpsd.fix.epv):
84                     self.last_epv = self.gpsd.fix.epv
85                 if not math.isnan(self.gpsd.fix.speed):
86                     self.last_speed = self.gpsd.fix.speed
87                 if not math.isnan(self.gpsd.fix.eps):
88                     self.last_eps = self.gpsd.fix.eps
89                 if self.gpsd.utc != None and self.gpsd.utc != '':
90                     self.last_time = self.gpsd.utc
91
92         logging.info('gps: terminating')
93

```

7.2.3.9 def burn.spec_proc.GpsThread.speed (self)

Definition at line 119 of file spec_proc.py.

```

119     def speed(self):
120         return self.last_speed
121

```

7.2.3.10 def burn.spec_proc.GpsThread.time (self)

Definition at line 127 of file spec_proc.py.

```

127     def time(self):
128         return self.last_time
129

```

7.2.4 Member Data Documentation

7.2.4.1 burn.spec_proc.GpsThread.gpsd

Definition at line 49 of file spec_proc.py.

7.2.4.2 `burn.spec_proc.GpsThread.last_alt`

Definition at line 54 of file `spec_proc.py`.

7.2.4.3 `burn.spec_proc.GpsThread.last_eps`

Definition at line 57 of file `spec_proc.py`.

7.2.4.4 `burn.spec_proc.GpsThread.last_epv`

Definition at line 55 of file `spec_proc.py`.

7.2.4.5 `burn.spec_proc.GpsThread.last_epx`

Definition at line 51 of file `spec_proc.py`.

7.2.4.6 `burn.spec_proc.GpsThread.last_epy`

Definition at line 53 of file `spec_proc.py`.

7.2.4.7 `burn.spec_proc.GpsThread.last_lat`

Definition at line 50 of file `spec_proc.py`.

7.2.4.8 `burn.spec_proc.GpsThread.last_lon`

Definition at line 52 of file `spec_proc.py`.

7.2.4.9 `burn.spec_proc.GpsThread.last_speed`

Definition at line 56 of file `spec_proc.py`.

7.2.4.10 `burn.spec_proc.GpsThread.last_time`

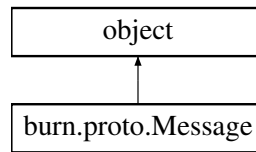
Definition at line 58 of file `spec_proc.py`.

The documentation for this class was generated from the following file:

- [/home/drb/dev/py/burn/spec_proc.py](#)

7.3 burn.proto.Message Class Reference

Inheritance diagram for burn.proto.Message:



Public Member Functions

- `def __init__(self, command="", arguments={})`

Public Attributes

- `command`
- `arguments`

7.3.1 Detailed Description

Class used to store a protocol message

Definition at line 20 of file proto.py.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `def burn.proto.Message.__init__(self, command = ' ', arguments = { })`

Initialize message

Definition at line 24 of file proto.py.

```
24     def __init__(self, command='', arguments={}):
25         """
26         Initialize message
27         """
28         self.command = command
29         self.arguments = arguments
30
```

7.3.3 Member Data Documentation

7.3.3.1 `burn.proto.Message.arguments`

Definition at line 29 of file proto.py.

7.3.3.2 burn.proto.Message.command

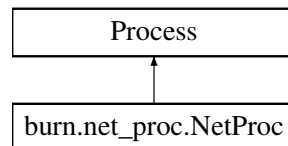
Definition at line 28 of file proto.py.

The documentation for this class was generated from the following file:

- </home/drb/dev/py/burn/proto.py>

7.4 burn.net_proc.NetProc Class Reference

Inheritance diagram for burn.net_proc.NetProc:



Public Member Functions

- `def __init__ (self, fd)`
- `def run (self)`
- `def dispatch_ctrl_msg (self, msg)`
- `def dispatch_net_msg (self)`
- `def is_running (self)`

Public Attributes

- `fd`
- `addr`
- `sock`
- `buffer`

7.4.1 Detailed Description

Definition at line 28 of file net_proc.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 def burn.net_proc.NetProc.__init__(self, fd)

Description:

Initialization of the net process

Arguments:

fd - File descriptor to send and receive messages to/from controller

Definition at line 30 of file net_proc.py.

```

30     def __init__(self, fd):
31         """
32         Description:
33             Initialization of the net process
34         Arguments:
35             fd - File descriptor to send and receive messages to/from controller
36         """
37         Process.__init__(self)
38         self.fd = fd
39         setblocking(self.fd, 0)
40         self._running = False
41         self.conn, self.addr = None, None
42         self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
43         self.sock.setblocking(0)
44         self.buffer = ''
45         try:
46             self.sock.bind((HOST, PORT))
47         except socket.error as e:
48             logging.error('network: bind failed: ' + os.strerror(e.errno))
49
50         self.sock.listen(5)
51         logging.info('network: service listening')
52

```

7.4.3 Member Function Documentation

7.4.3.1 def burn.net_proc.NetProc.dispatch_ctrl_msg(self, msg)

Description:

Serialize a controller message to a netstring and send it to ground control

Arguments:

msg - The controller message to dispatch

Definition at line 109 of file net_proc.py.

```

109    def dispatch_ctrl_msg(self, msg):
110        """
111        Description:
112            Serialize a controller message to a netstring and send it to ground control
113        Arguments:
114            msg - The controller message to dispatch
115        """
116        if msg.command == 'close_ok': # main controller is closing
117            self._running = False
118
119        data = ''
120        if msg.command == 'spectrum_ready': # Meta message
121            # 'spectrum_ready' is a meta message indicating that a response message is stored on disk
122            # The path to the file is stored in msg.arguments['filename']
123            with open(msg.arguments["filename"]) as jfd:
124                m = json.load(jfd) # Load json from file
125                data = json.dumps(m)
126        else: # Regular message
127            data = json.dumps(msg.__dict__) # Convert object to json
128
129        # Serialize the message into a netstring
130        netstring = struct.pack("!I", len(data)) + data
131        # Send the message over the network

```

```

132         totlen, currlen = len(netstring), 0
133         while True: # Continue until the full message is transferred
134             l = self.conn.send(netstring[currlen:])
135             if l == 0:
136                 inputs.remove(self.conn)
137                 self.conn.close()
138                 logging.info('network: connection broken from ' + self.addr[0])
139                 break
140             currlen += l
141             if currlen >= totlen:
142                 break
143

```

7.4.3.2 def burn.net_proc.NetProc.dispatch_net_msg (self)

Description:

Convert received data to messages and pass them to the controller

Definition at line 144 of file net_proc.py.

```

144     def dispatch_net_msg(self):
145         """
146         Description:
147             Convert received data to messages and pass them to the controller
148         """
149         while True:
150             if len(self.buffer) < 4:
151                 return
152             # Extract message length
153             msglen = struct.unpack("I", self.buffer[0:4])[0]
154             if len(self.buffer) < msglen+4:
155                 logging.info('network: buffer not ready')
156                 return
157             # Extract rest of message and convert to object
158             jmsg = json.loads(self.buffer[4:4+msglen])
159             msg = Message(**jmsg)
160             # Pass message to main controller
161             self.fd.send(msg)
162             # Update buffer
163             self.buffer = self.buffer[4+msglen:]
164

```

7.4.3.3 def burn.net_proc.NetProc.is_running (self)

Description:

Return wether the net process is still running

Definition at line 165 of file net_proc.py.

```

165     def is_running(self):
166         """
167         Description:
168             Return wether the net process is still running
169         """
170         return self._running
171

```

7.4.3.4 def burn.net_proc.NetProc.run (self)

Description:

Entry point for the net process

Definition at line 53 of file net_proc.py.

```

53     def run(self):
54         """
55         Description:
56             Entry point for the net process
57         """
58         logging.info('network: starting service')
59         self._running = True
60         # Prepare sockets and file descriptors
61         inputs = [self.fd, self.sock]
62
63         # Start select event loop
64         while(self._running):
65             readable, _, _ = select.select(inputs, [], [])
66
67             for s in readable: # Handle reads
68                 if s is self.sock: # Incoming connection on listening socket
69                     # We only allow one connection at a time (TODO)
70                     self.conn, self.addr = s.accept()
71                     self.conn.setblocking(0)
72                     inputs.append(self.conn)
73                     self.buffer = ''
74                     logging.info('network: connection received from ' + self.
addr[0])
75
76                 elif s is self.fd: # Incoming message from main controller
77                     self.dispatch_ctrl_msg(s.recv())
78
79                 else: # Incoming data from existing connection
80                     try:
81                         data = s.recv(1024)
82                     except socket.error as e:
83                         if e.errno == errno.ECONNRESET:
84                             # Unexpected disconnect from client
85                             inputs.remove(s)
86                             s.close()
87                             logging.error('network: ' + self.addr[0] + ': ' + os.strerror(e.errno))
88                             continue
89                         if not data or data == '':
90                             # Unexpected disconnect from client
91                             inputs.remove(s)
92                             s.close()
93                             logging.error('network: connection lost')
94                             continue
95                     else:
96                         # Data successfully received, store in buffer
97                         self.buffer += data
98                         self.dispatch_net_msg()
99
100         # Close active connections
101         if self.conn is not None:
102             self.conn.close()
103         if self.sock is not None:
104             self.sock.close()
105         self.fd.close()
106
107         logging.info('network: terminating')
108

```

7.4.4 Member Data Documentation

7.4.4.1 burn.net_proc.NetProc.addr

Definition at line 41 of file net_proc.py.

7.4.4.2 burn.net_proc.NetProc.buffer

Definition at line 44 of file net_proc.py.

7.4.4.3 `burn.net_proc.NetProc.fd`

Definition at line 38 of file `net_proc.py`.

7.4.4.4 `burn.net_proc.NetProc.sock`

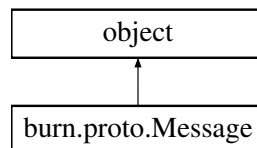
Definition at line 42 of file `net_proc.py`.

The documentation for this class was generated from the following file:

- [/home/drb/dev/py/burn/net_proc.py](#)

7.5 object Class Reference

Inheritance diagram for object:

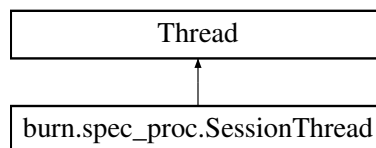


The documentation for this class was generated from the following file:

- [/home/drb/dev/py/burn/proto.py](#)

7.6 `burn.spec_proc.SessionThread` Class Reference

Inheritance diagram for `burn.spec_proc.SessionThread`:



Public Member Functions

- `def __init__` (self, event, target, msg)
- `def run` (self)

7.6.1 Detailed Description

Thread class to govern a single session

Definition at line 130 of file `spec_proc.py`.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 def burn.spec_proc.SessionThread.__init__(self, event, target, msg)

Description:

Initialize the session thread

Arguments:

event - Event to notify exit

target - Function running the detector

msg - The session message containing info about this session

Definition at line 134 of file spec_proc.py.

```

134     def __init__(self, event, target, msg):
135         """
136         Description:
137             Initialize the session thread
138         Arguments:
139             event - Event to notify exit
140             target - Function running the detector
141             msg - The session message containing info about this session
142         """
143         threading.Thread.__init__(self)
144         self._stopped = event
145         self._target = target
146         self._msg = msg
147

```

7.6.3 Member Function Documentation

7.6.3.1 def burn.spec_proc.SessionThread.run(self)

Description:

Entry point for the session thread

Definition at line 148 of file spec_proc.py.

```

148     def run(self):
149         """
150         Description:
151             Entry point for the session thread
152         """
153         logging.info('session: starting')
154         # Extract the session variables from the session message
155         delay = float(self._msg.arguments["delay"]) # Time to wait between each spectrum
156         iterations = int(self._msg.arguments["iterations"]) # Number of spectrums to take
157         infinite = iterations == -1 # If iterations is -1, run forever
158         index = 0 # Keep track of spectrums (spectrum id)
159
160         while not self._stopped.wait(delay):
161             if not infinite:
162                 # Exit when we reach a zero spectrum count
163                 iterations = iterations - 1
164                 if iterations < 0:
165                     break
166             # Run the detector
167             self._target(self._msg, index)
168             index = index + 1
169
170         logging.info('session: terminating')
171

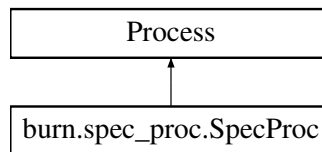
```

The documentation for this class was generated from the following file:

- /home/drb/dev/py/burn/spec_proc.py

7.7 burn.spec_proc.SpecProc Class Reference

Inheritance diagram for burn.spec_proc.SpecProc:



Public Member Functions

- `def __init__ (self, fd)`
- `def run (self)`
- `def send_msg (self, msg)`
- `def dispatch (self, msg)`
- `def stabilize_probe (self, voltage, coarse_gain, fine_gain)`
- `def run_acquisition_once (self, req_msg, session_index)`
- `def reset_acquisition (self)`
- `def run_acquisition (self, msg, session_index)`
- `def save_acquisition (self, msg, session_index)`
- `def save_acquisition_as_chn (self, sd, msg, session_index)`

Public Attributes

- [fd](#)
- [running](#)
- [send_lock](#)
- [gps_stop](#)
- [gps_client](#)
- [group](#)
- [input](#)
- [dtb](#)
- [session_stop](#)
- [session](#)

7.7.1 Detailed Description

Process class for handling the gps and spectrometry

Definition at line 172 of file `spec_proc.py`.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 def burn.spec_proc.SpecProc.__init__(self, fd)

Description:

Initialize the spectrometer process

Arguments:

fd - file descriptor to pass and receive messages to/from controller

Definition at line 176 of file spec_proc.py.

```

176     def __init__(self, fd):
177         """
178         Description:
179             Initialize the spectrometer process
180         Arguments:
181             fd - file descriptor to pass and receive messages to/from controller
182         """
183         Process.__init__(self)
184         self.fd = fd
185         self.running = False
186         self.send_lock = threading.Lock() # Lock used to synchronize sending of messages to
controller
187         self.gps_stop = threading.Event() # Event used to notify gps thread
188         self.gps_client = GpsThread(self.gps_stop) # Create the gps thread
189         # Initialize the detector
190         self.group = 1
191         self.input = 1
192         self.dtb = DeviceFactory.createInstance(DeviceFactory.DeviceInterface.IDevice)
193         self.dtb.open("", Utilities.getLynxAddress())
194         logging.info('spec: using device ' + self.dtb.getParameter(ParameterCodes.Network_MachineName, 0))
195         self.dtb.lock("administrator", "password", self.input)
196

```

7.7.3 Member Function Documentation

7.7.3.1 def burn.spec_proc.SpecProc.dispatch(self, msg)

Description:

Handle a message received from controller

Arguments:

msg - The message received

Definition at line 234 of file spec_proc.py.

```

234     def dispatch(self, msg):
235         """
236         Description:
237             Handle a message received from controller
238         Arguments:
239             msg - The message received
240         """
241         if msg.command == 'set_gain': # Set the gain parameters for the detector
242             voltage = msg.arguments["voltage"]
243             coarse = msg.arguments["coarse_gain"]
244             fine = msg.arguments["fine_gain"]
245             self.stabilize_probe(voltage, coarse, fine)
246             logging.info('spec: gain has been set')
247             msg.command = 'set_gain_ok' # Notify ground control that gain has been set
248             self.send_msg(msg)
249         elif msg.command == 'close': # Controller wants us to close down
250             self.running = False
251         elif msg.command == 'new_session': # Start a new session
252             msg.command = 'new_session_ok'
253             self.send_msg(msg)
254             self.session_stop = threading.Event()
255             self.session = SessionThread(self.session_stop, self.
run_acquisition_once, msg)
256             self.session.start()

```

```

257         elif msg.command == 'stop_session': # Stop any running sessions
258             if not self.session_stop.isSet():
259                 self.session_stop.set()
260                 self.session.join()
261                 logging.info('spec: session stopped')
262             msg.command = 'stop_session_ok' # Notify ground control that we have stopped any sessions
263             self.send_msg(msg)
264         else:
265             # Unknown command received from controller
266             logging.warning('spec: unknown command ' + cmd.command)
267

```

7.7.3.2 def burn.spec_proc.SpecProc.reset_acquisition (self)

Description:

Reset and initialize the detector

Definition at line 340 of file spec_proc.py.

```

340     def reset_acquisition(self):
341         """
342         Description:
343             Reset and initialize the detector
344         """
345         #Disable all acquisition
346         Utilities.disableAcquisition(self.dtb, self.input)
347         #Set the acquisition mode. The Only Available Spectral in Osprey is Pha = 0
348         self.dtb.setParameter(ParameterCodes.Input_Mode, 0, self.input)
349         #Setup presets
350         self.dtb.setParameter(ParameterCodes.Preset_Options, 1, self.input)
351         #Clear data and time
352         self.dtb.control(CommandCodes.Clear, self.input)
353         #Set the current memory group
354         self.dtb.setParameter(ParameterCodes.Input_CurrentGroup, self.group, self.
input)
355

```

7.7.3.3 def burn.spec_proc.SpecProc.run (self)

Description:

Entry point for the spectrometer process

Definition at line 197 of file spec_proc.py.

```

197     def run(self):
198         """
199         Description:
200             Entry point for the spectrometer process
201         """
202         logging.info('spec: staring service')
203         self.running = True
204
205         self.gps_client.start() # Start the gps
206
207         # Event loop
208         while(self.running):
209             if self.fd.poll():
210                 self.dispatch(self.fd.recv()) # Handle messages from the controller
211
212         # Cleanup and exit
213         self.fd.close()
214         self.gps_stop.set()
215         self.gps_client.join()
216         logging.info('spec: GPS client stopped')
217         logging.info('spec: terminating')
218

```


7.7.3.4 def burn.spec_proc.SpecProc.run_acquisition (self, msg, session_index)

Description:

Run the detector

Arguments:

msg - The response message

session_index - The sequence number in current session

Definition at line 356 of file spec_proc.py.

```

356     def run_acquisition(self, msg, session_index):
357         """
358         Description:
359             Run the detector
360         Arguments:
361             msg - The response message
362             session_index - The sequence number in current session
363         """
364         # Setup presets
365         livetime = float(msg.arguments["livetime"])
366         self.dtb.setParameter(ParameterCodes.Preset_Live, livetime, self.input)
367         # Clear data and time
368         self.dtb.control(CommandCodes.Clear, self.input)
369         # Start the acquisition
370         self.dtb.control(CommandCodes.Start, self.input)
371         while True:
372             sd = self.dtb.getSpectralData(self.input, self.group)
373             if ((0 == (StatusBits.Busy & sd.getStatus())) and (0 == (StatusBits.Waiting & sd.getStatus()))):
374                 break
375             time.sleep(.1)
376
377         # Extract last spectrum from detector and prepare parameters
378         chans = sd.getSpectrum().getCounts()
379         total_count = 0
380         channel_string = ''
381         for ch in chans:
382             total_count += ch
383             channel_string += str(ch) + ' '
384
385         # Add spectrum data to the response message
386         msg.arguments["channels"] = channel_string.strip()
387         msg.arguments["channel_count"] = len(chans)
388         msg.arguments["uncorrected_total_count"] = total_count
389         msg.arguments["livetime"] = sd.getLiveTime()
390         msg.arguments["realtime"] = sd.getRealTime()
391         msg.arguments["computational_limit"] = sd.getComputationalValue()
392         msg.arguments["spectral_input"] = sd.getInput()
393         msg.arguments["spectral_group"] = sd.getGroup()
394         msg.arguments["spectral_status"] = Utilities.getStatusDescription(sd.getStatus())
395

```

7.7.3.5 def burn.spec_proc.SpecProc.run_acquisition_once (self, req_msg, session_index)

Description:

Gather info from gps and detector

Arguments:

req_msg - The session message

session_index - The sequence number in current session

Definition at line 293 of file spec_proc.py.

```

293     def run_acquisition_once(self, req_msg, session_index):
294         """
295         Description:
296             Gather info from gps and detector
297         Arguments:
298             req_msg - The session message
299             session_index - The sequence number in current session
300         """
301         # Prepare the response message

```

```

302     resp_msg = copy.deepcopy(req_msg)
303     resp_msg.command = 'spectrum'
304     resp_msg.arguments['session_index'] = session_index
305
306     # Reset detector
307     self.reset_acquisition()
308
309     # Gather gps info before running the detector
310     resp_msg.arguments['latitude_start'] = self.gps_client.latitude
311     resp_msg.arguments['latitude_start_err'] = self.gps_client.epx
312     resp_msg.arguments['longitude_start'] = self.gps_client.longitude
313     resp_msg.arguments['longitude_start_err'] = self.gps_client.epy
314     resp_msg.arguments['altitude_start'] = self.gps_client.altitude
315     resp_msg.arguments['altitude_start_err'] = self.gps_client.epv
316     resp_msg.arguments['gps_speed_start'] = self.gps_client.speed
317     resp_msg.arguments['gps_speed_start_err'] = self.gps_client.eps
318     resp_msg.arguments['gps_time_start'] = self.gps_client.time
319
320     # Run the detector
321     self.run_acquisition(resp_msg, session_index)
322
323     # Gather gps info after running the detector
324     resp_msg.arguments['gps_time_end'] = self.gps_client.time
325     resp_msg.arguments['latitude_end'] = self.gps_client.latitude
326     resp_msg.arguments['latitude_end_err'] = self.gps_client.epx
327     resp_msg.arguments['longitude_end'] = self.gps_client.longitude
328     resp_msg.arguments['longitude_end_err'] = self.gps_client.epy
329     resp_msg.arguments['altitude_end'] = self.gps_client.altitude
330     resp_msg.arguments['altitude_end_err'] = self.gps_client.epv
331     resp_msg.arguments['gps_speed_end'] = self.gps_client.speed
332     resp_msg.arguments['gps_speed_end_err'] = self.gps_client.eps
333
334     # Save acquisition to file and send a meta message to controller
335     fn = self.save_acquisition(resp_msg, session_index)
336     m = Message('spectrum_ready')
337     m.arguments["filename"] = fn
338     self.send_msg(m)
339

```

7.7.3.6 def burn.spec_proc.SpecProc.save_acquisition (self, msg, session_index)

Description:

Save the gps and specter data to file (json format)

Arguments:

msg - The response message

session_index - The sequence number in current session

Definition at line 396 of file spec_proc.py.

```

396     def save_acquisition(self, msg, session_index):
397         """
398         Description:
399             Save the gps and specter data to file (json format)
400         Arguments:
401             msg - The response message
402             session_index - The sequence number in current session
403         """
404         # Build the path to store the response message
405         session_name = msg.arguments['session_name']
406         session_dir = os.path.expanduser("~/ashes/") + session_name
407         if not os.path.isdir(session_dir):
408             os.makedirs(session_dir, 0777)
409         fname = session_dir + os.path.sep + str(session_index) + ".json"
410         # Store the response message to file
411         with open(fname, "w") as f:
412             json.dump(msg.__dict__, f)
413         return fname
414

```

7.7.3.7 def burn.spec_proc.SpecProc.save_acquisition_as_chn (self, sd, msg, session_index)**Description:**

Save the the specter data to file (chn format)

Arguments:

sd - Spectrum data
 msg - The session message
 session_index - The sequence number in current session

Definition at line 415 of file spec_proc.py.

```

415     def save_acquisition_as_chn(self, sd, msg, session_index):
416         """
417         Description:
418             Save the the specter data to file (chn format)
419         Arguments:
420             sd - Spectrum data
421             msg - The session message
422             session_index - The sequence number in current session
423         """
424         session_name = msg.arguments['session_name']
425         session_dir = os.path.expanduser("~/ashes/") + session_name
426         if not os.path.isdir(session_dir):
427             os.makedirs(session_dir, 0777)
428         chans = sd.getSpectrum().getCounts()
429         mca, sec, rt, lt, dat, tim, off, nc = 1, 0, sd.getRealTime(), sd.getLiveTime(), "07DEC151", "0707",
0, len(chans) # FIXME
430         hdr = pack("hhhhi8s4shh", -1, mca, 1, sec, rt, lt, dat, tim, off, nc)
431         with open(session_dir + os.path.sep + str(session_index) + ".chn", "w+b") as f:
432             f.write(hdr)
433             int_array = array('L', chans)
434             int_array.tofile(f)
435

```

7.7.3.8 def burn.spec_proc.SpecProc.send_msg (self, msg)**Description:**

Function to safely pass messages to controller

Arguments:

msg - The message to pass

Definition at line 219 of file spec_proc.py.

```

219     def send_msg(self, msg):
220         """
221         Description:
222             Function to safely pass messages to controller
223         Arguments:
224             msg - The message to pass
225         """
226         self.send_lock.acquire()
227         try:
228             self.fd.send(msg)
229         except:
230             logging.error('spec: send exception: ' + sys.exc_info()[0])
231         finally:
232             self.send_lock.release()
233

```

7.7.3.9 `def burn.spec_proc.SpecProc.stabilize_probe (self, voltage, coarse_gain, fine_gain)`

Description:

Set gain parameters for the detector

Arguments:

voltage - The voltage level

coarse_gain - The coarse gain level

fine_gain - The fine gain level

Definition at line 268 of file `spec_proc.py`.

```

268     def stabilize_probe(self, voltage, coarse_gain, fine_gain):
269         """
270         Description:
271             Set gain parameters for the detector
272         Arguments:
273             voltage - The voltage level
274             coarse_gain - The coarse gain level
275             fine_gain - The fine gain level
276         """
277         # Osprey API constants
278         Stabilized_Probe_Bussy = 0x00080000
279         Stabilized_Probe_OK = 0x00100000
280         dtb_probe_type = self.dtb.getParameter(ParameterCodes.Input_Status, self.
input)
281         # Set voltage
282         if((dtb_probe_type & Stabilized_Probe_OK) != Stabilized_Probe_OK):
283             self.dtb.setParameter(ParameterCodes.Input_Voltage, int(voltage), self.
input)
284             self.dtb.setParameter(ParameterCodes.Input_VoltageStatus, True, self.
input)
285             # Wait till ramping is complete
286             logging.info('spec: ramping HVPS...')
287             while(self.dtb.getParameter(ParameterCodes.Input_VoltageRamping, self.
input) is True):
288                 time.sleep(.4)
289             # Set coarse and fine gain
290             self.dtb.setParameter(ParameterCodes.Input_CoarseGain, float(coarse_gain), self.
input) # [1.0, 2.0, 4.0, 8.0]
291             self.dtb.setParameter(ParameterCodes.Input_FineGain, float(fine_gain), self.
input) # [1.0, 5.0]
292

```

7.7.4 Member Data Documentation

7.7.4.1 `burn.spec_proc.SpecProc.dtb`

Definition at line 192 of file `spec_proc.py`.

7.7.4.2 `burn.spec_proc.SpecProc.fd`

Definition at line 184 of file `spec_proc.py`.

7.7.4.3 `burn.spec_proc.SpecProc.gps_client`

Definition at line 188 of file `spec_proc.py`.

7.7.4.4 `burn.spec_proc.SpecProc.gps_stop`

Definition at line 187 of file `spec_proc.py`.

7.7.4.5 burn.spec_proc.SpecProc.group

Definition at line 190 of file spec_proc.py.

7.7.4.6 burn.spec_proc.SpecProc.input

Definition at line 191 of file spec_proc.py.

7.7.4.7 burn.spec_proc.SpecProc.running

Definition at line 185 of file spec_proc.py.

7.7.4.8 burn.spec_proc.SpecProc.send_lock

Definition at line 186 of file spec_proc.py.

7.7.4.9 burn.spec_proc.SpecProc.session

Definition at line 255 of file spec_proc.py.

7.7.4.10 burn.spec_proc.SpecProc.session_stop

Definition at line 254 of file spec_proc.py.

The documentation for this class was generated from the following file:

- [/home/drb/dev/py/burn/spec_proc.py](#)

Chapter 8

File Documentation

8.1 /home/drb/dev/py/burn/__init__.py File Reference

Namespaces

- [burn](#)

8.2 /home/drb/dev/py/burn/burn.py File Reference

Classes

- class [burn.burn.Burn](#)

Namespaces

- [burn.burn](#)

Variables

- [burn.burn.filename](#)
- [burn.burn.level](#)

8.3 /home/drb/dev/py/burn/helpers.py File Reference

Namespaces

- [burn.helpers](#)

Functions

- def [burn.helpers.setblocking](#) (fd, state)

8.4 /home/drb/dev/py/burn/net_proc.py File Reference

Classes

- class [burn.net_proc.NetProc](#)

Namespaces

- [burn.net_proc](#)

Variables

- string [burn.net_proc.HOST](#) = "
- int [burn.net_proc.PORT](#) = 7000

8.5 /home/drb/dev/py/burn/proto.py File Reference

Classes

- class [burn.proto.Message](#)

Namespaces

- [burn.proto](#)

8.6 /home/drb/dev/py/burn/README.md File Reference

8.7 /home/drb/dev/py/burn/spec_proc.py File Reference

Classes

- class [burn.spec_proc.GpsThread](#)
- class [burn.spec_proc.SessionThread](#)
- class [burn.spec_proc.SpecProc](#)

Namespaces

- [burn.spec_proc](#)

Index

- [/home/drb/dev/py/burn/README.md, 40](#)
- [/home/drb/dev/py/burn/__init__.py, 39](#)
- [/home/drb/dev/py/burn/burn.py, 39](#)
- [/home/drb/dev/py/burn/helpers.py, 39](#)
- [/home/drb/dev/py/burn/net_proc.py, 40](#)
- [/home/drb/dev/py/burn/proto.py, 40](#)
- [/home/drb/dev/py/burn/spec_proc.py, 40](#)
- [__enter__](#)
 - [burn::burn::Burn, 16](#)
- [__exit__](#)
 - [burn::burn::Burn, 16](#)
- [__init__](#)
 - [burn::burn::Burn, 15](#)
 - [burn::net_proc::NetProc, 25](#)
 - [burn::proto::Message, 23](#)
 - [burn::spec_proc::GpsThread, 19](#)
 - [burn::spec_proc::SessionThread, 29](#)
 - [burn::spec_proc::SpecProc, 31](#)
- [addr](#)
 - [burn::net_proc::NetProc, 27](#)
- [altitude](#)
 - [burn::spec_proc::GpsThread, 19](#)
- [arguments](#)
 - [burn::proto::Message, 23](#)
- [buffer](#)
 - [burn::net_proc::NetProc, 27](#)
- [burn, 11](#)
- [burn.burn, 11](#)
- [burn.burn.Burn, 15](#)
- [burn.helpers, 12](#)
- [burn.net_proc, 12](#)
- [burn.net_proc.NetProc, 24](#)
- [burn.proto, 13](#)
- [burn.proto.Message, 23](#)
- [burn.spec_proc, 13](#)
- [burn.spec_proc.GpsThread, 18](#)
- [burn.spec_proc.SessionThread, 28](#)
- [burn.spec_proc.SpecProc, 30](#)
- [burn::burn](#)
 - [filename, 11](#)
 - [level, 11](#)
- [burn::burn::Burn](#)
 - [__enter__, 16](#)
 - [__exit__, 16](#)
 - [__init__, 15](#)
 - [dispatch_net_msg, 16](#)
 - [dispatch_spec_msg, 17](#)
 - [fdn, 17](#)
 - [fds, 17](#)
 - [n, 18](#)
 - [run, 17](#)
 - [running, 18](#)
 - [s, 18](#)
- [burn::helpers](#)
 - [setblocking, 12](#)
- [burn::net_proc](#)
 - [HOST, 12](#)
 - [PORT, 12](#)
- [burn::net_proc::NetProc](#)
 - [__init__, 25](#)
 - [addr, 27](#)
 - [buffer, 27](#)
 - [dispatch_ctrl_msg, 25](#)
 - [dispatch_net_msg, 26](#)
 - [fd, 27](#)
 - [is_running, 26](#)
 - [run, 26](#)
 - [sock, 28](#)
- [burn::proto::Message](#)
 - [__init__, 23](#)
 - [arguments, 23](#)
 - [command, 23](#)
- [burn::spec_proc::GpsThread](#)
 - [__init__, 19](#)
 - [altitude, 19](#)
 - [eps, 19](#)
 - [epv, 20](#)
 - [epx, 20](#)
 - [epy, 20](#)
 - [gpsd, 21](#)
 - [last_alt, 21](#)
 - [last_eps, 22](#)
 - [last_epv, 22](#)
 - [last_epx, 22](#)
 - [last_epy, 22](#)
 - [last_lat, 22](#)
 - [last_lon, 22](#)
 - [last_speed, 22](#)
 - [last_time, 22](#)
 - [latitude, 20](#)
 - [longitude, 20](#)
 - [run, 20](#)
 - [speed, 21](#)
 - [time, 21](#)
- [burn::spec_proc::SessionThread](#)
 - [__init__, 29](#)
 - [run, 29](#)

- burn::spec_proc::SpecProc
 - `__init__`, 31
 - `dispatch`, 31
 - `dtb`, 36
 - `fd`, 36
 - `gps_client`, 36
 - `gps_stop`, 36
 - `group`, 36
 - `input`, 37
 - `reset_acquisition`, 32
 - `run`, 32
 - `run_acquisition`, 32
 - `run_acquisition_once`, 33
 - `running`, 37
 - `save_acquisition`, 34
 - `save_acquisition_as_chn`, 34
 - `send_lock`, 37
 - `send_msg`, 35
 - `session`, 37
 - `session_stop`, 37
 - `stabilize_probe`, 35
- command
 - burn::proto::Message, 23
- dispatch
 - burn::spec_proc::SpecProc, 31
- dispatch_ctrl_msg
 - burn::net_proc::NetProc, 25
- dispatch_net_msg
 - burn::burn::Burn, 16
 - burn::net_proc::NetProc, 26
- dispatch_spec_msg
 - burn::burn::Burn, 17
- dtb
 - burn::spec_proc::SpecProc, 36
- eps
 - burn::spec_proc::GpsThread, 19
- epv
 - burn::spec_proc::GpsThread, 20
- epx
 - burn::spec_proc::GpsThread, 20
- epy
 - burn::spec_proc::GpsThread, 20
- fd
 - burn::net_proc::NetProc, 27
 - burn::spec_proc::SpecProc, 36
- fdn
 - burn::burn::Burn, 17
- fds
 - burn::burn::Burn, 17
- filename
 - burn::burn, 11
- gps_client
 - burn::spec_proc::SpecProc, 36
- gps_stop
 - burn::spec_proc::SpecProc, 36
- gpsd
 - burn::spec_proc::GpsThread, 21
- group
 - burn::spec_proc::SpecProc, 36
- HOST
 - burn::net_proc, 12
- input
 - burn::spec_proc::SpecProc, 37
- is_running
 - burn::net_proc::NetProc, 26
- last_alt
 - burn::spec_proc::GpsThread, 21
- last_eps
 - burn::spec_proc::GpsThread, 22
- last_epv
 - burn::spec_proc::GpsThread, 22
- last_epx
 - burn::spec_proc::GpsThread, 22
- last_epy
 - burn::spec_proc::GpsThread, 22
- last_lat
 - burn::spec_proc::GpsThread, 22
- last_lon
 - burn::spec_proc::GpsThread, 22
- last_speed
 - burn::spec_proc::GpsThread, 22
- last_time
 - burn::spec_proc::GpsThread, 22
- latitude
 - burn::spec_proc::GpsThread, 20
- level
 - burn::burn, 11
- longitude
 - burn::spec_proc::GpsThread, 20
- n
 - burn::burn::Burn, 18
- object, 28
- PORT
 - burn::net_proc, 12
- reset_acquisition
 - burn::spec_proc::SpecProc, 32
- run
 - burn::burn::Burn, 17
 - burn::net_proc::NetProc, 26
 - burn::spec_proc::GpsThread, 20
 - burn::spec_proc::SessionThread, 29
 - burn::spec_proc::SpecProc, 32
- run_acquisition
 - burn::spec_proc::SpecProc, 32
- run_acquisition_once
 - burn::spec_proc::SpecProc, 33
- running

burn::burn::Burn, [18](#)
burn::spec_proc::SpecProc, [37](#)

s

burn::burn::Burn, [18](#)
save_acquisition
 burn::spec_proc::SpecProc, [34](#)
save_acquisition_as_chn
 burn::spec_proc::SpecProc, [34](#)
send_lock
 burn::spec_proc::SpecProc, [37](#)
send_msg
 burn::spec_proc::SpecProc, [35](#)
session
 burn::spec_proc::SpecProc, [37](#)
session_stop
 burn::spec_proc::SpecProc, [37](#)
setblocking
 burn::helpers, [12](#)
sock
 burn::net_proc::NetProc, [28](#)
speed
 burn::spec_proc::GpsThread, [21](#)
stabilize_probe
 burn::spec_proc::SpecProc, [35](#)
time
 burn::spec_proc::GpsThread, [21](#)